Klaus Kohl-Schoepe

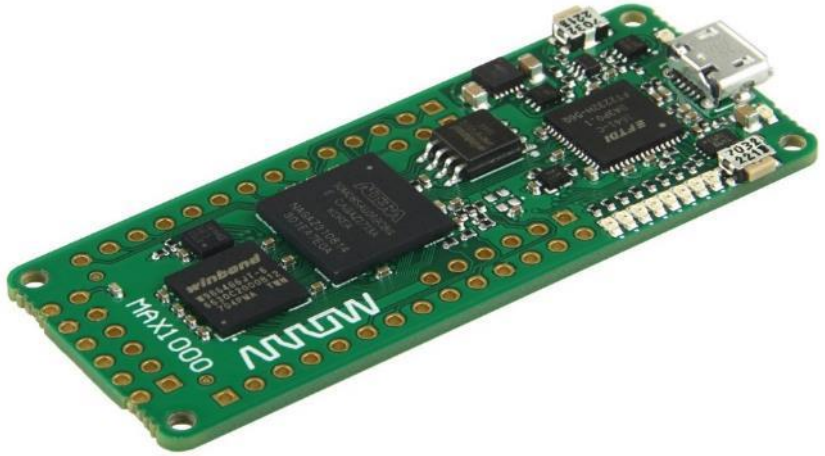
August 13, 2022

Using low cost FPGA boards for FORTH

Klaus Kohl-Schoepe
kks@designin.de
www.mcforth.net

MAX1000

... the IoT Maker Solution!

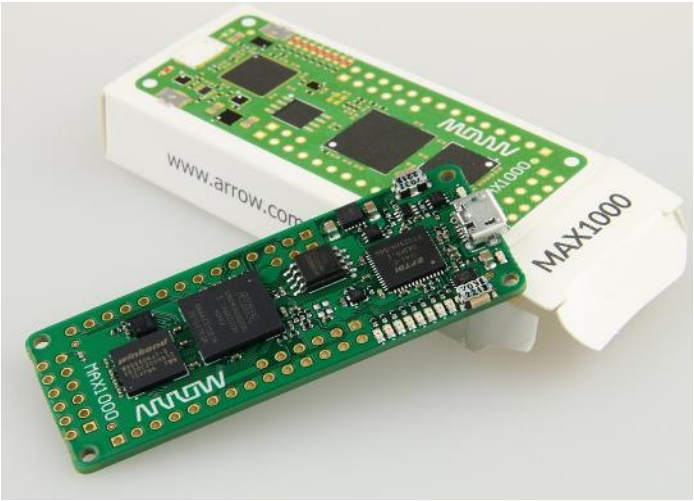


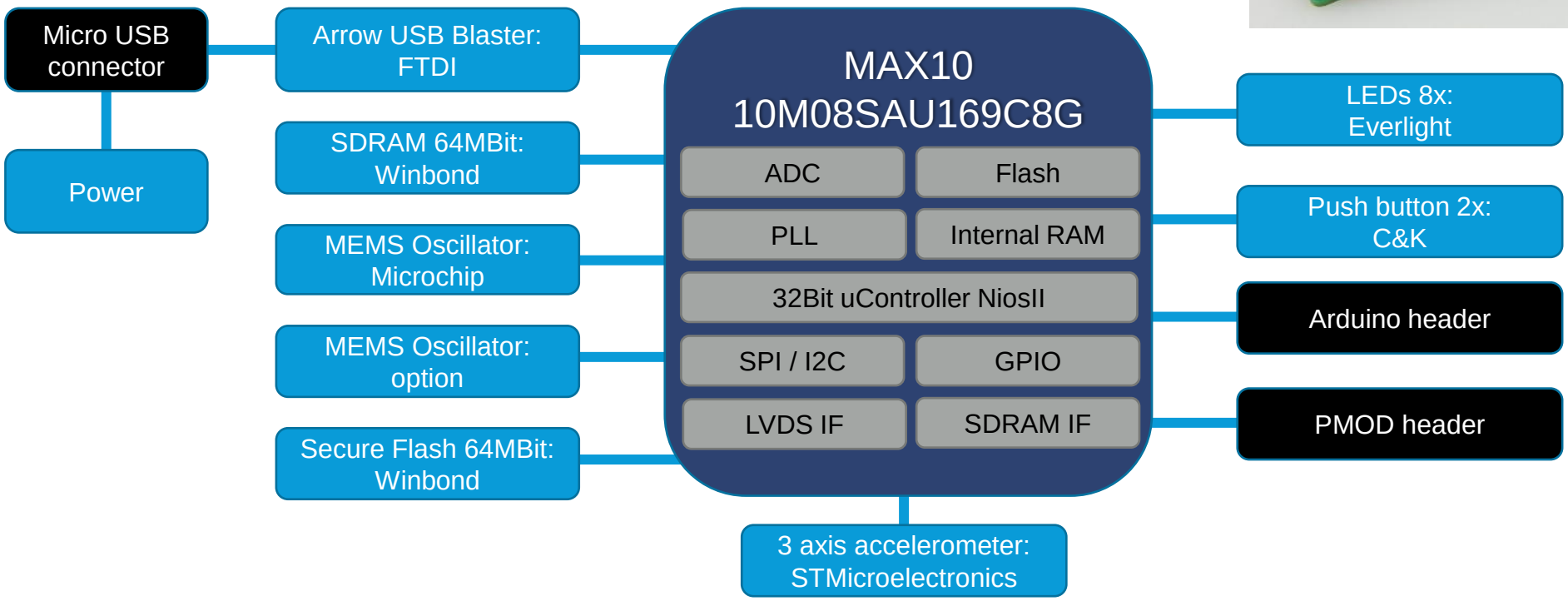
Feature Set:

- > Lowest cost MAX10 solution on the market
- > Intel MAX10 with 8kLE
- > Arduino MKR standard 25x61.5mm²
- > Integrated Arrow USB Blaster
- > Preprogrammed Demo Application
- > Plug&play full featured FPGA kit
- > PMOD connector to adapt various solutions
- > Comes in an attractive box
- > **Qualified hardware also for real end products in a customized version!!!**

MAX1000

... Block diagram





CYC1000

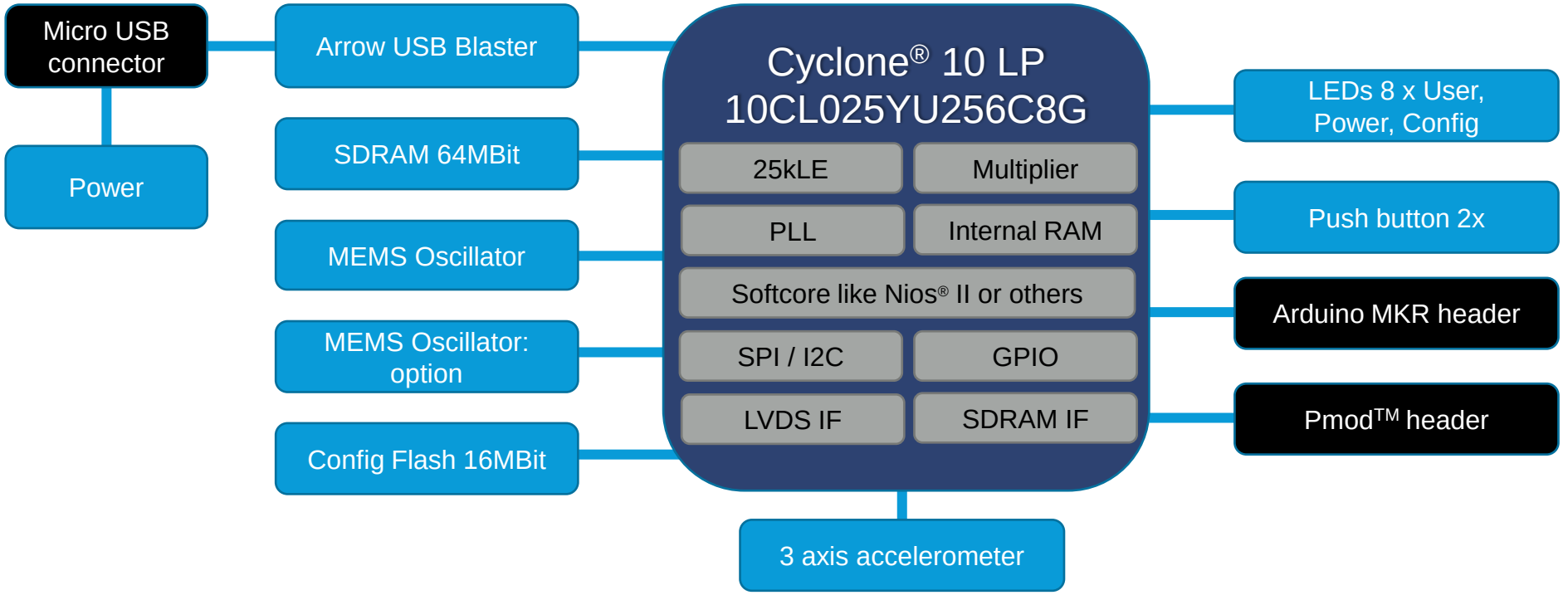
... the IoT Maker Solution!

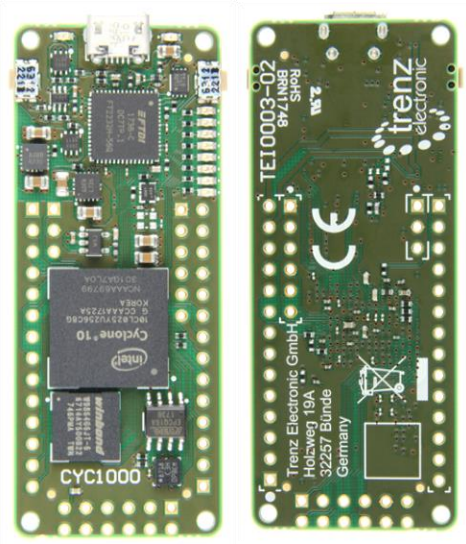
Feature Set:

- > One of the lowest cost Cyclone® 10 LP solution
- > Arduino MKR standard 25x61.5mm²
- > Intel® Cyclone® 10 LP with
  - 25k Logic Elements (6k/10k/16k possible with UBGA256)
  - 66 M9K Memory (594kb)
  - 66 18*18-bit Multiplier
- > Integrated Arrow USB Blaster
- > Preprogrammed Demo Application
- > Plug&play full featured FPGA kit
- > **Qualified hardware also for real end products in a customized version!!!**

CYC1000

... Block diagram





Other small FPGA Boards from Arrow

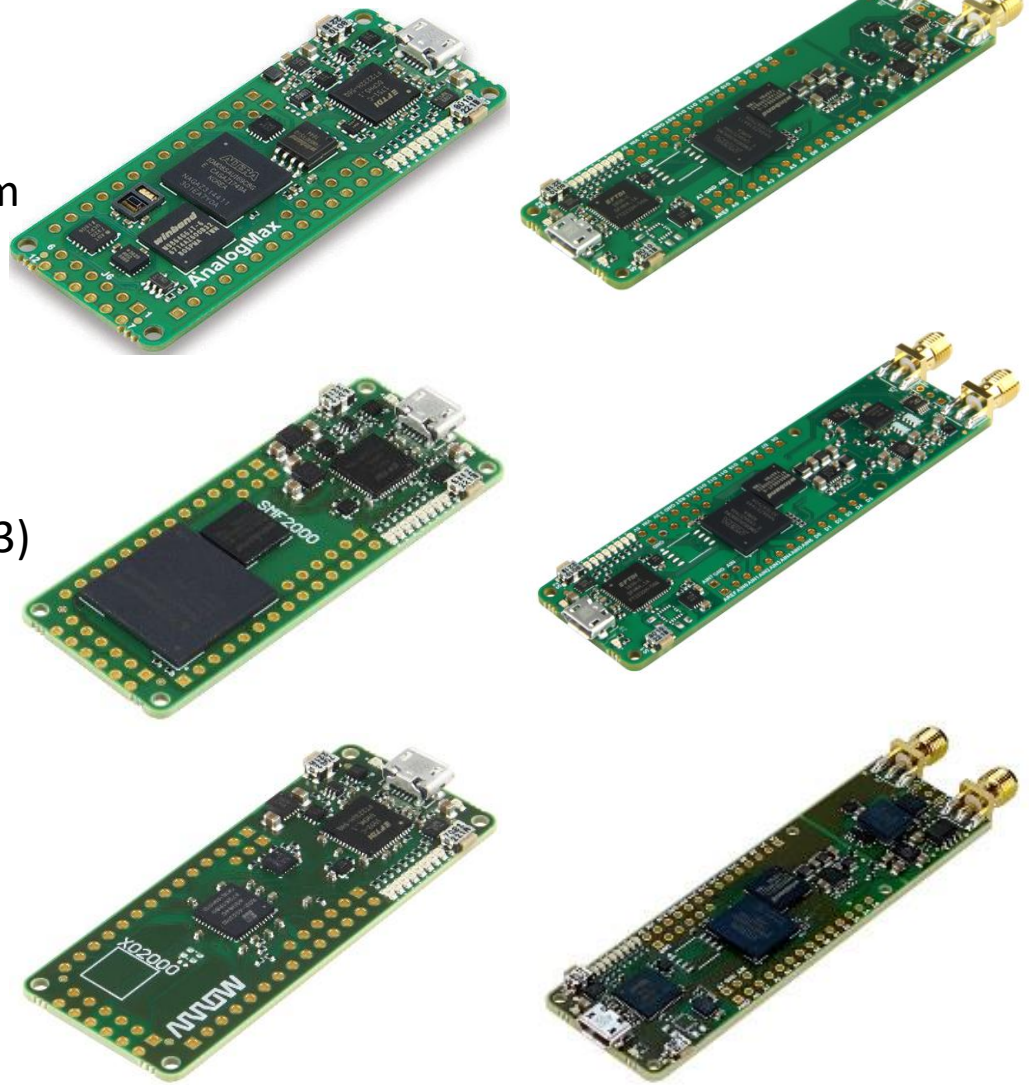
Same form factor – different applications

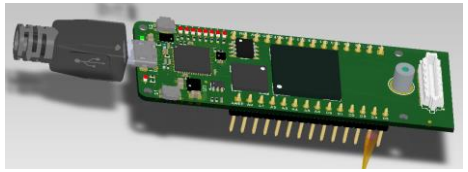
- AnalogMax-01 –Programmable Sensor Fusion Development Platform
- AnalogMax-DAQ1 - with AD4003 (18-bit, 1MSPS)
- AnalogMax-DAQ2 - with ADAQ7980 (16-bit, 1MSPS)
- AnalogMax-DAQ2-500k - with ADAQ7988 (16-bit, 500kSPS)
- AnalogMax-DAQ3 - with ADAQ4003 (18-bit, 2MSPS)

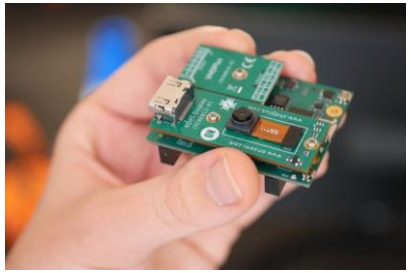
- SMF2000 - Microchip SmartFusion 2 Board (with 150MHz Cortex-M3)
- LXO2000 - Lattice MACHXO2 (4kLE)

In Development:

- CYC5000 - with Cyclone® V E (25kLE) and CRUVI HS connector
- …



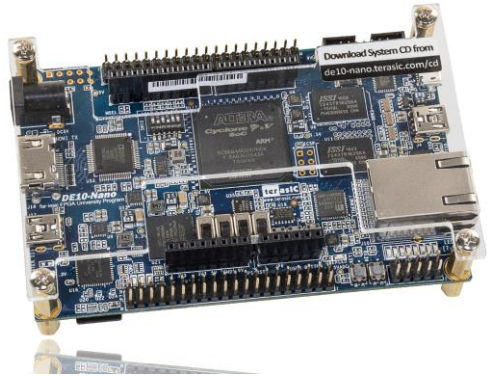




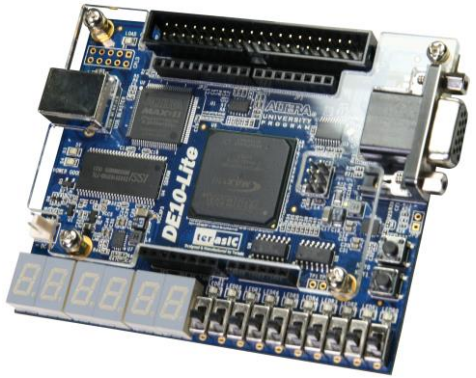
Other low cost FPGA Boards

From Intel, Lattice, and Xilinx

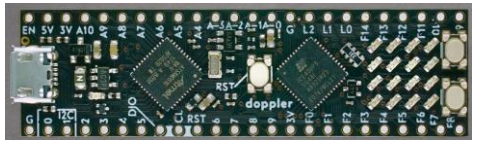
- Intel (Terasic):
  - DE10-Nano or Arrow SoCKit (Cyclone V SoC – dual core A9)
  - DE10-Lite Board (MAX10 – 50kLE and VGA)



- Lattice:
  - MachXO2 Pico Development Kit (1200LE, 1Mb SPI Flash)
  - Lattice XP2 Brevia2 Development Kit (used from Ting for EP32)
  - Dada Machine Doppler (SAMD51 and ICE40)



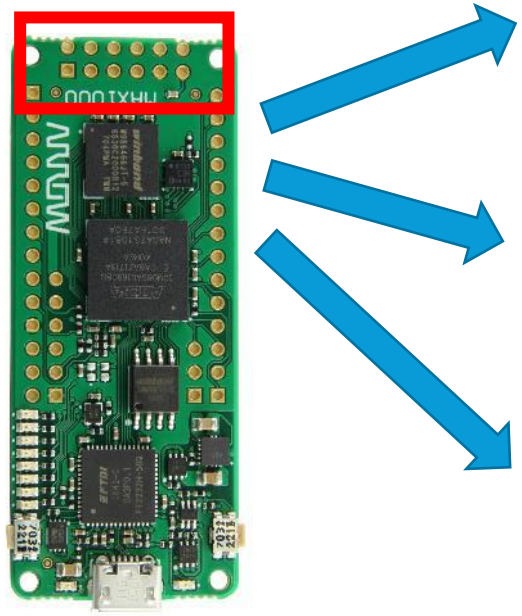
- Xilinx
  - Papilion (Spartan 3E) – not longer available but was used e.g. for J1
  - Trenz TE0722 Zynq-7000 Board (dual A9)



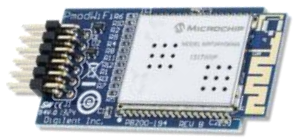
Connectivity 1

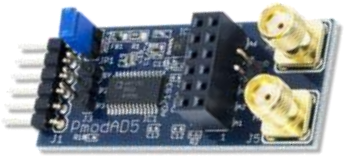
... PMOD connector... build easily more applications!

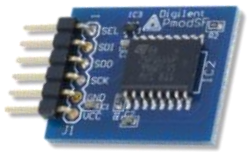
PMOD Standard Connector



PMOD Adapter Boards







Application

Wifi / BLE
RTC

24Bit ADC
3 axis sensor

9 axis sensor
Flash

....and many more!!!

Supplier Solution

MICROCHIP

ANALOG DEVICES

life.augmented

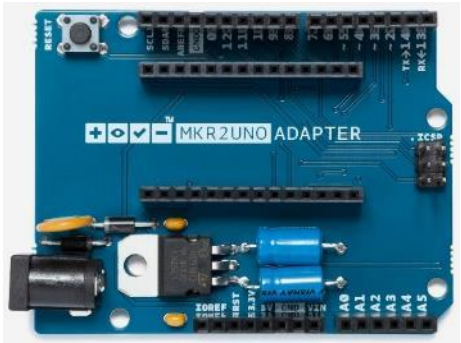
Connectivity 2

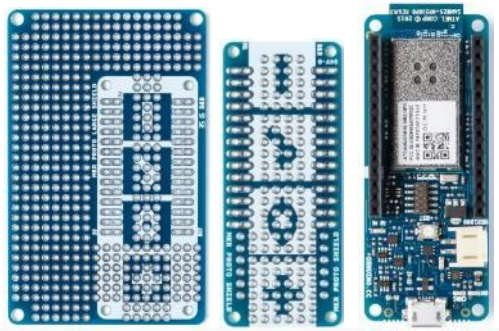
... Arduino features... build easily various applications

Arduino
MKR Standard

Arduino
MKR2UNO Adapter

Arduino
Shields





Ideas for Applications and Expansions

Using other Processors, Languages and IDE‘s

- Arduino
  - Direct update of software over serial interface possible
  - Normally you need an AVR (MegaAVR) core to use the original software framework
  - A lot of adaption are made to use the same framework under “normal” C environment
  - Maybe an adapter from MKR to Arduino is helpful

- Other processor cores are possible
  - ARM Cortex-M can be critical because of licenses
  - RISC V are available in different flavors fitting in <<8kLE (even multicore)
  - 8051, 6502, Z80, PIC, AVR and 6808 are often used in old projects so adaption of these software is easier
  - A lot of old video games using cores like 1802 or 6502 adaptable to this board (e.g. Pong)
  - Some programming language like JAVA or Python can be accelerated with own cores

- Own Cores and Language
  - FORTH (K1 and SmallForth – need only a terminal and editor to develop a program)
  - BASIC (also now seldom used)
  - MicroPython (actual in focus but no free version available)

Standard Processor IP for FPGA

Many processor IP's (~222) are available at http://opencores.org

- Processor IP's (>200) on http://opencores.org
  - AVR and PIC16
  - 8051, 8080, 8085, and Z80
  - 6502
  - 6805, 6808, 6809 and 6305, 6811
  - 8086, 80186, or 80486
  - RISC-V
  - Lot of other 8-32bit (RISC) processors (1802, MSP430, …)
  - even FORTH processor with Java optimization

- FPGA/Forth projects on https://github.com
  - 6809, J1, N.I.G.E together with (FIG-)Forth (see also article from Bernd Paysan – FORTH Tagung 2016)
  - … nearly the same as above

- FPGA platform for emulation of old video games and "Personal Computer" linke VC20, C64, Tandy …
  - MISTer FPGA Project – use Cyclone V SoC (DE10-Nano, SoCKit) for simulation of NES, Game Boy, C64, …

Forth Processor IP

Mostly realized on one of the mentioned boards above

- James Bowman's J1 Forth CPU with SWAPFORTH or complete system H2 using eForth
- C. H. Ting's EP16 (for Cyclone IV) or EP32 (Lattice XP2) and eForth for PDP1 or 8080 on FPGA
- Don Golding's older P16 or actual concept phase of FP1 (CORE I - work in progress)
- Brad Eckert's CD16 and MSL16
- Phil Koopman CPU/16
- Richard E. Haskell's FP16 Forth Core
- Bernd Paysan's B16
- MPE's RTXcore (RTX-2000 für FPGA)
- Klaus Scheisik's MicroCore
- Douglas W. Jones MISC16 with eForth from Steve Teal
- Mixed Mode RISC-1 (supported by Jürgen Pintaske und Jens Wilke but no source available)
- Andrew Read's N.I.G.E.-Machine

- And some concepts:
  - Philip J. Koopman Jr.'s WISC Concept
  - Richard E. Haskell and Darring M. Hanna W8Z
  - Lennard Benschop SOD32 (Simulator)

My FORTH and FPGA activities

So many boards and so little time – see mcforth.net (German)

- KKForth – my first implementations before 30 years for x86 /V20, Z80, and RTX2000 (1990)
- S8Forth – a 8K Forth for the Zilog Super8 ROM (1990)
- mcForth – my own FORTH implementation (VP32-Simulator in 386 assembler) and XMC1xxx (Cortex-M0) (2002…)
- Adaption of J1 FORTH IP to my K1 core and realization of SmallFORTH (see next slides) for MAX1000 (2020)
- K1 for CYC1000 (2020)
- First tests with 6502 core on MAX1000 (2020)
- K1 for DE10-Lite Kit (also MAX10) with VGA interface (2021)
- Getting a Longan Nano Kit leads to RISC-V assembler, disassembler, and simulator using mcForth (2021)
- First tests with RISC-V and mecrisp on MAX1000 (2022)
- VP32 (the opcode behind mcForth) runs on MAX1000 (2022)
- Test of FIG-Forth using 1802 processor IP on MAX1000 (2022)

Why K1 (instead of J1)

I like 8-Bit/Byte microcontroller

- Memory organized in 8-bit per address (32KByte available on MAX1000)
- Opcode and variable aligned to 16-bit (2 byte)
- Using High-Endian (easier to read in hex dump)
- Literals using ±16k instead 0-32k (bit15=bit14; inv14 toggle bit14)
- Calls use absolute address (/2 => 0..32766), Jumps relative offset (/2 => ±4KByte)
- Additional jump used for FOR … NEXT loop (jump and decrement if TOR<>0)
- Arithmetic during read and write of memory

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	value															Literal (0..32K)
0	0	0	target													Jump
0	0	1	target													Jump of T=0
0	1	0	target													Call
0	1	1	;	ALU				>N	>R	!		rr		ss		Arithmetic

J1 opcode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	literal															Literal (±16k)
1	0	addr														Call
1	1	0	0	relativ												Jump
1	1	0	1	relativ												Jump if T=0
1	1	1	0	relativ												Jump if R<>0
1	1	1	1	0	;	>R	>N	ALU				BW	@!	ss		Memory access
1	1	1	1	1	;	>R	>N	ALU				rr		ss		Arithmetik

K1 opcode

Why SmallForth

I use always own target compiler written in Forth

- Long time experience in writing assembler, disassembler, and simulator for different processors
- I like to use a nearly standard FORTH like F83 or ANS – maybe reduced to minimum
- I use a terminal (Tera Term) for downloading the application of smaller programs on FPGA
- Difficult to handle upper case letter on german keyboard so I ignore it

This leads to a compact Forth (~6.5KByte with K1 on MAX1000 or CYC1000):

```
init 'init quit find words .s dump Abort" ?Abort" ?abort abort 'abort postpone ['] ' name> >name body> >body
\IFNDEF \IFDEF \IF \ELSE \THEN ENDCASE ENDOF OF CASE NEXT ?FOR FOR BEGIN WHILE UNTIL REPEAT AGAIN ELSE THEN IF
AHEAD recurse Does> Create Variable Constant ; : :noname Header forget indirect restrict immediate ." s" [char]
char Literal ] [ call, $, , c, align allot $>number? >number . .r u. u.r 0u.r d. d.r decimal hex #> sign #s
# hold <# \\ \ ( -parse parse refill source upc accept type cr spaces space loadu saveu f_dump f_c@ f_c! f_ess
f_es ms key key? emit emit? button? f_data f_baud acc_data acc_baud pmod_dir pmod_data counter_hi counter_lo leds
uart_data uart_state uart_baud 2@ 2! count ! @ c! c@ dabs d- d+ / m/mod um/mod * m* um* nop negate invert xor
or and cells chars flip du2/ d2/ d2* rshift ashift lshift u2/ 2/ 2* within max min abs u< < <> = 0<> 0= 0<
aligned cell+ char+ 2- 1- 2+ 1+ - + goto execute ?exit exit rdrop ra> r> r@ >r rclear rdepth ?dup -rot rot
2drop 2swap 2over 2dup nip drop swap tuck over dup dclear depth span >in hld dpl base last state here voc-link
#ramstart #ramend #tib #c/tib true false bl #msb #bits
```

`'init` and `'abort` are variables used to define own reset or error handler.
Hardware functions for buttons, key/emit (UART), accelerometer (SPI), PMOD interface, counter, and leds.
No DO … LOOP because n FOR … R@ … NEXT are much faster (looping n times).
With saveu the (autostart) program will be saved and used, if no buttons are pressed during reset.

Life demonstration

Using MAX1000 and CYC1000

- Quartus Development Environment
- Steve Teal's MISC16/eFORTH on MAX1000
- SmallForth on MAX1000 and CYC1000
- Example program downloads:
  - ANS tester
  - LIFE on terminal
  - Accelerator readout
- Mecrisp using J1 on DE10-Lite Kit

Thank you

Questions?

Klaus Kohl-Schoepe
kks@designin.de
www.mcforth.net